gTFRC : a QoS-aware congestion control algorithm

Guillaume Jourjon, Emmanuel Lochin, Laurent Dairaine

*g*TFRC : a QoS-aware congestion control algorithm

Emmanuel Lochin
National ICT Australia Ltd,
Locked Bag 9013,
Alexandria, NSW 1435
Australia

Laurent Dairaine
National ICT Australia Ltd,
ENSICA - LAAS/CNRS,
1, place Emile Blouin
31056 Toulouse Cedex 5
France

Guillaume Jourjon
National ICT Australia Ltd,
ENSICA - LAAS/CNRS,
1, place Emile Blouin
31056 Toulouse Cedex 5
France

{emmanuel.lochin,laurent.dairaine, guillaume.jourjon}@nicta.com.au

Abstract

This study addresses the end-to-end congestion control support over the DiffServ Assured Forwarding (AF) class. The resulting Assured Service (AS) provides a minimum level of throughput guarantee. In this context, this paper describes a new end-to-end mechanism for continuous transfer based on TCP-Friendly Rate Control (TFRC) originally proposed in [11]. The proposed approach modifies TFRC to take into account the QoS negotiated. This mechanism, named gTFRC, is able to reach the minimum throughput guarantee whatever the flow's RTT and target rate. Simulation measurements show the efficiency of this mechanism either in over-provisioned or exactly-provisioned network. In addition, we show that the gTFRC mechanism can be used in the same DiffServ/AF class with TCP or TFRC flows.

1 Introduction

The increasing capabilities of high performance end systems and communication networks have greatly accelerated the development of distributed computing. Distributed applications were originally characterized by very basic communication requirements mainly related to full reliability and order. Today, many applications are demanding more complex requirements in terms of delay and bandwidth. In the context of Internet, the Assured Forwarding class of the IETF/DiffServ [12] provides a guaranteed minimal throughput that many applications can take advantage of. The service offered is called Assured Service (AS) and built on top of the AF PHB. The minimum assured throughput (also called target rate) is given according to a negotiated profile with the user. Such traffic is generated by adaptive applications, it means that their throughput can increase as long as there are available resources and can decrease when a congestion occurs.

Nevertheless, this QoS support is alone not sufficient to cope with either the application requirements (e.g., reliability, timing) or the network control requirements. Most of the today's Internet applications use TCP [23] as a mean to transport their data. TCP offers a reliable and in sequence end-to-end stream oriented data transfer service between two interconnected systems. Moreover, TCP implements flow and congestion control mechanisms in order to avoid receivers' buffers overflowing and network congestion. Despite of a TCP good behavior in terms of available bandwidth sharing, TCP is not appropriate for many applications that integrates time and bandwidth constraints. The TCP-friendly Rate Control (TFRC) [11] is an equation based congestion control mechanism operating in the best-effort Internet environment and competing with TCP traffic. TFRC has a much lower variation of throughput over time than TCP. As a result, it is more suitable for multimedia application such as video streaming or VoIP.

This paper focus on how TFRC mechanism behaves in the context of a DiffServ/AF class. We show by simulation the good properties of classical TFRC in terms of bandwidth smoothing and sharing when it is mixed with either other TFRC or TCP flows. Nevertheless, even if a throughput guarantee is provided to the application by the underlying network, as for TCP, the throughput obtained by TFRC mainly depends on RTT and loss probability. Then, the application does not always get the negotiated guaranteed throughput also called the target rate. To cope with this problem, we propose a very simple adaptation to TFRC, namely *g*TFRC, allowing the application to reach its target rate whatever the RTT value of the application's flow. Results from a large simulation campaign is presented to demonstrate the improvements and to exhibit the TCP-

friendliness of gTFRC in various situations.

This paper is structured as follow. The section 2 presents the context of this study and provides a related work about the DiffServ/AF and congestion control mechanisms. Section 3 gives the problem statement and presents the gTFRC mechanism. Section 4 evaluates gTFRC and finally section 5 provides a conclusion.

2 Background

2.1 Context of this study : the EuQoS project

Many research works have been carried out on Quality of Service mechanisms for packet switching networks over the past ten years. The results of these efforts have still not been transformed into a large multi-domain network providing QoS guarantees [1]. Without loss of generality, this study takes place in the context of the EuQoS project. The EuQoS project [3] is an integrated project under the European Union's Framework Program 6 which aims at deploying a flexible and secure QoS assurance system over a pan-European testbed environment. The EuQoS System integrates many applications requiring QoS guarantees such as Voice over IP, Video on Demand or Medical applications over multi-domain heterogeneous environment such as WiFi, UMTS, xDSL or Ethernet technologies. Then, the EuQoS system integrates various architectural components such as signaling protocols, traffic engineering mechanisms, QoS routing, admission control to resource reservation scheme and of course multimedia application and transport protocols.

2.2 Related work

There have been a number of studies that focused on assured service for TCP flows. In [25], five factors have been studied (RTT, number of flows, target rate, packet size, non responsive flows) and their impact has been evaluated in providing a predictable service for TCP flows. In an over-provisioned network, target rates are achieved regardless of these five factors. This result is corroborated by [9]. However, the distribution of the excess bandwidth depends on these five factors. Recently [21] demonstrates the unfair allocation of out-of-profile TCP traffic and concludes that the aggregate that has the smaller/larger target rate, occupies more/less bandwidth than its fair-share regardless of the subscription level. As the TCP protocol uses the AIMD control congestion algorithm which tries to share fairly the available bandwidth, the only meaning to obtain a service differentiation with TCP protocol is to use DiffServ traffic conditioners such token bucket color marker [13] or time sliding window color marker [5]. The behavior of the traffic conditioner has a great impact on the service level, in terms of bandwidth, obtained by TCP flows. Several others conditioners have been proposed to improve throughput insurance [2], [7], [10], [16], [17], [18], [20]. In these articles, it is clearly shown that the key of the problem is the valued trio $(loss\_probability, RTT, target\_rate)$ of a TCP flow. At our best knowledge, there is a only few studies of TFRC behavior in a DiffServ network. In particular, [14] investigates AF-TFRC performances and gives a service provisioning mechanism allowing an ISP to build a feasible DiffServ system. In this study, the problem of high RTT difference between long and short transfer are not tackled. Moreover, for experimentations purposes (based on loss rate estimation), all simulations are carried during 1000 seconds. So, this duration allows a TFRC flow to converge easily to the target rate and, as a result, obtain an average throughput near the target rate.

3 gTFRC : a QoS-aware rate congestion control

3.1 Problem statement

As related in section 2.2, the only way to obtain a service differentiation with TCP protocol is to use DiffServ traffic conditioners. Indeed, the AIMD principle do not use the instantaneous TCP throughput as an input value for its congestion control. On the contrary, TFRC congestion protocol uses this value in conjunction with the RTT and the loss event of the flow (this value is symbolized in the gTFRC 's name by the letter g). These values are used in order to give a smooth rate adaptation to the flow and are efficient for streaming media applications that do not require absolute reliability. Assume that an adaptive application uses TFRC. This application uses the AF DiffServ class and negotiate a target rate. In the assured service class, the throughput of these flows breaks up into two parts:

1. a fixed part which corresponds to a minimum assured throughput. In the event of congestion in the network, the packets of this part are marked like inadequate for loss (colored green or marked in-profile);

2. an elastic part which corresponds to an opportunist flow of packets (colored red or marked out-profile). No guarantee is brought to these packets. They are conveyed by the network on the principle of "best-effort" (BE) and are dropped first if a congestion occurs.

We assume that the network is well-provisioned and that the whole in-profile traffic does not exceed the resource allocated to the AF class. In case of excess bandwidth in the network, the application could send more than its target rate, so the network should mark its excess traffic out-of-profile. If the network becomes congestioned, many out-of-profile

losses occur and the optimal rate estimated by TFRC could be under the target rate requested by the application. TCP would react in a same manner by halving its congestion window. As for TCP in the AF class, the TFRC flow is not aware that the loss is corresponding to an out-of-profile packet and that it should not decrease its target rate. For TCP, the solution was to find a conditioner able to mark better the TCP flows that the simple token bucket as explained in section 2.2, or propose to add a new QoS congestion window as in [6] or [26]. In order to avoid this case and allow to TFRC to send always under its target rate, we propose to make the sending rate estimator aware of the target rate. This is achieved by computing the sending rate as the maximum between the TFRC rate estimation and the target rate. Thanks to this knowledge, the application's flow is sent in conformance with the negotiated QoS while staying TCP-friendly in its out-profile part.

3.2 Implementation

In the previous section, we saw that *g*TFRC requires the knowledge of the bandwidth guarantee the DiffServ/AF network service provides to the session. We assume it is configured at socket creation time, directly by the application. Without loss of generality, this parameter is supposed to be known by application after it has been previously negotiated in an end-to-end basis by the way of proper signalization protocol that an architecture like EuQoS System [3] provides. In order to evaluate the proposition, we use the TFRC code issue from ns-2 simulator. Figure 1 gives the modification we made on the TFRC's code. The lines beginning by a "+" show the modification applied in the rate estimation function. We can see on this algorithm that this maximum value is not applied during the TFRC slow-start period in order to stay friendly with the others TFRC or TCP flows.

4 Evaluation and analysis

*g*TFRC is evaluated over a DiffServ network using simulation. We use ns-2.28 simulator and the Nortel DiffServ implementation [22]. We drive simulation on the testbed illustrated in the figure 2 with the two following scenarios: when the network is exactly-provisioned (when there is no excess bandwidth for the out-profile traffic) and when the network is over-provisioned (when there is excess bandwidth). As a network under-provisioned (in this case, the amount of in-profile traffic is higher than the resource allocated to the AF class) is considered as a wrong dimensioned network, we exclude this case.

```
if (first_pkt_rcvd == 0) {
    first_pkt_rcvd = 1 ;
    slowstart();
    nextpkt();
}
else {
      if (rate_change_ == SLOW_START) {
         if (flost > 0) {
             rate_change_ = OUT_OF_SLOW_START;
             oldrate_ = rate_ = rcvrate;
             } else {
                    slowstart();
                    nextpkt();
                   }
         } else {
+                 if (rcvrate > gtfrc_)
+                 {
                    if (rcvrate > rate_)
                        increase_rate(flost);
                    else
                        decrease_rate ();
+                } else {
+                        rate_ = gtfrc_;
+                        rate_change_ = CONG_AVOID;
+                        last_change_ = now;
+                        heavyrounds_ = 0;
+                }
         }
}
```

Figure 1. gTFRC algorithm

4.1 Simulation model and hypothesis

An important known problem in a DiffServ network is the unfair bandwidth sharing of the out-profile part. In [21], the authors provide a way to solve this problem; they show that if a network is exactly-provisioned, there is no bias in favor of a flow or an aggregate that has a smaller target rate. Taking this as the starting point, they infer that the unfairness problem can be solved by making the networks exactly-provisioned by simply adjusting the target rates of the token bucket marker in order to get a proportional differentiation. This assertion is strongly closed to the RTT and the loss probability of the network. So, in a first part of our experiments, we measure the behavior of TFRC and *g*TFRC

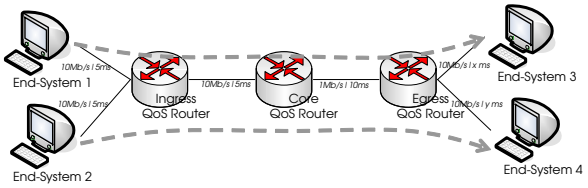


Figure 2. The simulation topology.

in order to evaluate the gain brought by gTFRC in this network case. In a second part, we made measurements with an over-provisioned network. It is important to note that we are not interested in finding the best conditioning mechanism in order to improve throughput insurance. This problem should be tackled by the network provider. In our case, we try to give a QoS transport protocol in conformance with the QoS negotiated in the network.

In all simulations:

- packet size is fixed to 1500 bytes;
- TCP version used is NewReno;
- we use a two color token bucket marker with a bucket size of 10^4 bytes;
- the queues size are 50 packets and RIO parameters are: $(min_{out}, max_{out}, p_{out}, min_{in}, max_{in}, p_{in}) = (10, 20, 0.1, 20, 40, 0.02)$;
- the bottleneck between the core and the egress router is $1000Kbits/s$;
- measurements are carried during $100sec$.

For each experiment, we evaluate the throughput at the server side and draw on the figures the instantaneous throughput and the cumulative average throughput. In a DiffServ multi-domain network, such the EuQoS network, RTTs difference between flows could be very large. Indeed, some flows can cross one or several DiffServ domains and obtain low or high RTT. Figure 3 illustrates this case. On this figure, the flow emitted between both end-system $A1$ and $A2$ should have an RTT higher than the flow emitted between $A1$ and $C1$. In order to take in consideration this case, we have chosen to compare flows with an high RTT difference (i.e., $600ms$).

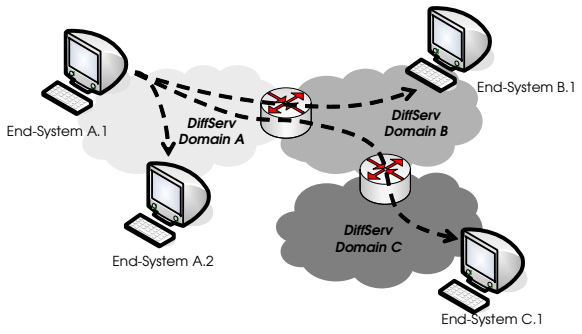


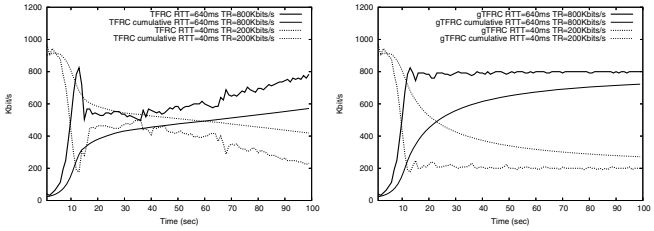
Figure 3. DiffServ multi-domain

4.2 Experiments in an exactly-provisioned network

4.2.1 Comparison between TFRC and gTFRC

We made experiments with many different RTTs and target rates configuration and give in this part a representative measurement of the efficiency of gTFRC . We measure the performance obtained by gTFRC in two scenarios. On figures 4, two flows are emitted on the testbed. The first one has a non favorable conditions since it has the highest target rate to reach and a high RTT $(RTT = 640ms, TR = 800Kbits/s)$. The second flow has the lowest target rate $(200Kbits/s)$ and a low RTT $(40ms)$.

The results for TFRC are presented on figure 4 (a) and for gTFRC on figure 4 (b). We can see that gTFRC allows to reach the target rate more quickly than with TFRC. The reason is obvious since at the first rate decrease evaluation of the algorithm, gTFRC evaluates a rate equal to the target rate. On figure 4 (a), we can see that this case of decreasing occurs for TFRC at $t = 11sec$ and that gTFRC does not calculate a rate lower than the negotiated target rate on figure 4 (b). Figure 4 (b) shows that the flow with the lowest target rate and the lowest RTT is constrained to reach its own target rate of $200Kbits/s$.

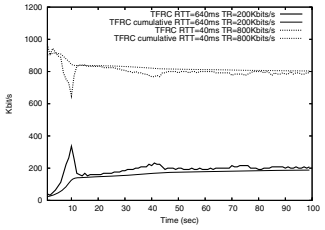


(a) TFRC only (b) gTFRC only

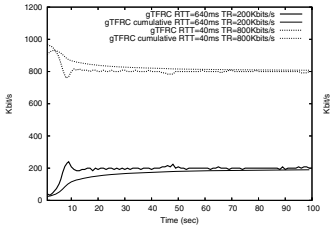
Figure 4. Exactly-provisioned network

In the second experiment, the target rates are reversed. This experiment shows that the gTFRC algorithm keeps the same behavior than TFRC in ideal condition. Figures 5 present the results obtained and show that in case of using gTFRC , the instantaneous throughput stays nearer to the target rate than TFRC.

Table 1 presents the packets statistics gathered during the simulation on figures 4 with `ldrops` for packets dropped due to link overflow and `edrops` for RED dropping. We can see that gTFRC increases the number of in-profile packets in the network. As a result, gTFRC gives an average throughput for each flow near their target rate since there are less out-profile packets in the network. These two measurements allow to conclude that gTFRC is DiffServ compliant and allow to get a throughput guarantee in the standard



(a) TFRC only



(b) *g*TFRC only

Figure 5. Exactly-provisioned network

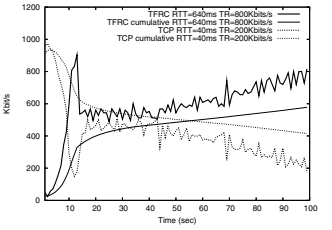
Code Point	Total packets	ldrops	edrops
in-profile	76.63%	0.02%	0%
out-profile	23.37%	3.41%	3.78%

(a) Packets statistics for scenario on figure 4 (a)

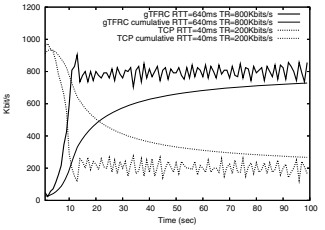
Code Point	Total packets	ldrops	edrops
in-profile	91.51%	0.01%	0.03%
out-profile	8.48%	12.66%	2.14%

(b) Packets statistics for scenario on figure 4 (b)

Table 1. Packets statistics for figure 4



(a) TFRC versus TCP



(b) *g*TFRC versus TCP

Figure 6. Exactly-provisioned network

Code Point	Total packets	ldrops	edrops
in-profile	73.57%	0.02%	0%
out-profile	26.42%	1.67%	3.12%

(a) Packets statistics for scenario on figure 6 (a)

Code Point	Total packets	ldrops	edrops
in-profile	90.16%	0.16%	0%
out-profile	9.84%	89.23%	2.84%

(b) Packets statistics for scenario on figure 6 (b)

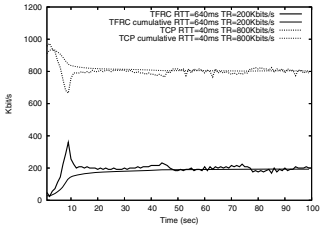
Table 2. Packets statistics for figure 6

DiffServ AF class in case of exactly-provisioned network. Moreover, a simple token bucket color marker seems to be able to characterize *g*TFRC flows.

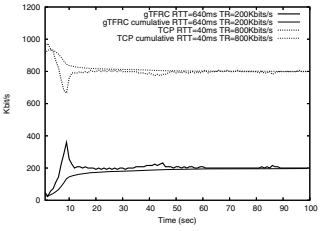
4.2.2 Comparison with TCP

As the DiffServ/AF class has been designed for elastic flows and in particular for TCP flows, we propose in this part to compare *g*TFRC and TCP flows together. TFRC has been built in order to be TCP-friendly. A flow is considered TCP-friendly or TCP-compatible when its long-term throughput does not exceed the throughput of a conformant TCP connection under the same conditions [8]. In case of a DiffServ network, this behavior can change due to the network conditioning. Indeed, the aim of a DiffServ network is to operate a differentiation between flows and not to share in a fair manner the bandwidth. As *g*TFRC is a specific mechanism for DiffServ network, it is not TCP-friendly in its in-profile part but it must remain TCP-friendly in its out-profile part. In order to show if *g*TFRC stays TCP-friendly in the out-profile part, we propose to determine if a similar behavior is obtained when a TCP flow is mixed with either a TFRC flow or a *g*TFRC flow. In order to verify this point, we keep the same network scenario than in the previous part but we compare TFRC and *g*TFRC with a TCP flow. Results are presented on figures 6 and 7.

If we compare with the previous experiments made on figures 4 and 5, we can see that the behavior of the TCP flow is similar to the TFRC flow and that the long-term throughput has the same order of magnitude. Table 2 presents the packets statistics obtained during the simulation on figures 6. These statistics are similar to these related on table 1. Nevertheless, we can see on table 2 (b) that the `ldrops` value is very high. This is due to the aggressive nature of TCP which tries to always reach a better throughput. This explains the oscillation of both figures 6. On figure 6 (b), since it has the best condition to release it (lower target rate and lower RTT), TCP tries to outperform its target rate and as a result, generates a high number of out-profile packets.



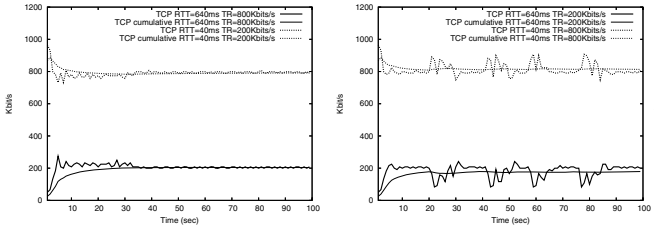
(a) TFRC versus TCP



(b) *g*TFRC versus TCP

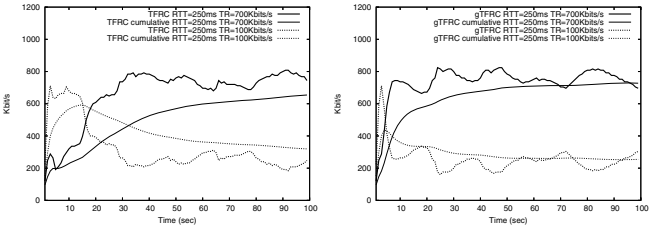
Figure 7. Exactly-provisioned network

Finally, to complete this comparison, we remind the well-known results obtained by two TCP flows in a DiffServ network on figures 8. We can see that TCP does not reach



(a) TCP only (b) TCP only

Figure 8. Exactly-provisioned network with TCP



(a) TFRC only (b) gTFRC only

Figure 9. Over-provisioned network (equal RTTs)

its target rate even if the network is exactly-provisioned in case of high RTT difference. On figure 8 (a), the flow which requests a target rate of $800Kbits/s$ obtains a throughput around $200Kbits/s$. If the highest RTT flow requests the lowest target rate, we can see on figure 8 (b) that this flow has difficulties to reach its target rate and that it oscillates under this value. In [24] and [19], analytical studies show that it is not always possible to achieve service differentiation with a simple token bucket in certain conditions. This example illustrates the case where the token bucket parameters has no effect for TCP flows on the desired target rate as raised in these studies. If we compare with figures 6 and 7, these results allow to verify that the use of gTFRC and TFRC flows in the same DiffServ class is not prejudicial for the TCP flows.

4.3 Experiments in an over-provisioned network

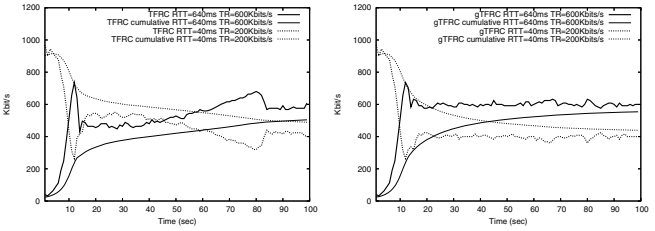
This part deals with the case of an over-provisioned network. We present measurement where the network let 20% of unallocated bandwidth. We investigate the case of various RTTs and different target rates.

4.3.1 Impact of the target rate

In this experiment, we focus on the influence of the negotiated target rate for TFRC and gTFRC . Both flows have an RTT equal to $250ms$ and two different target rates of $700Kbits/s$ and $100Kbits/s$. Figure 9 shows the results obtained with two flows with the same RTT but different target rates. We can see that gTFRC reaches its target rate quicker than TFRC. The explanation is the same than in section 4.2.1: if the throughput returned by the TFRC equation is lower to the target rate due to the loss event evaluation, gTFRC uses the target rate as optimal rate.

4.3.2 Impact of the RTT

In these experiments, we switch between two target rates of $600Kbits/s$ and $200Kbits/s$ and two RTTs of $640ms$ and $40ms$. On figures 10, one flow is in the worst condition to reach its target rate. It has the highest RTT and target rate. On figure 10 (a), we can see that the flow with the lowest RTT and target rate ($200Kbits/s$ and $40ms$) outperforms its target rate and that the other flow ($600Kbits/s$ and $640ms$) does not reach its target rate. So, as demonstrated in [24] and [19] for TCP flows, it seems that the token bucket marker is not a good traffic descriptor for the TFRC flows too. On the other hand, figure 10 (b) shows that gTFRC allows to enforce the desired target rate and that the gTFRC flow is able to reach its target rate. Table 3 gives packets statistics for this experiment and shows that gTFRC increases the number of in-profile packets in the network.



(a) TFRC only (b) gTFRC only

Figure 10. Over-provisioned network

Figures 11 deal with the reverse case and show that gTFRC has the same behavior than TFRC in ideal case.

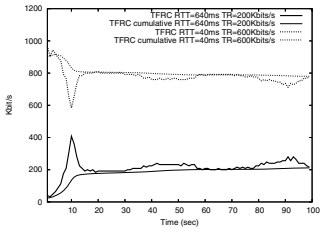
Code Point	Total packets	ldrops	edrops
in-profi le	67.90%	0.02%	0%
out-profi le	32.01%	0.36%	3.11%

(a) Packets statistics for scenario on fi gure 10 (a)

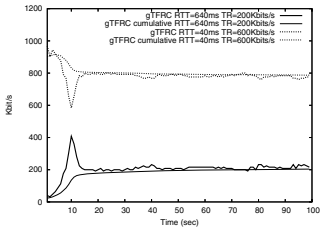
Code Point	Total packets	ldrops	edrops
in-profi le	74.71%	0.02%	0%
out-profi le	25.29%	0.47%	3.66%

(b) Packets statistics for scenario on fi gure 10 (b)

Table 3. Packets statistics for 10



(a) TFRC only

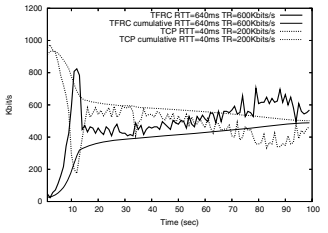


(b) gTFRC only

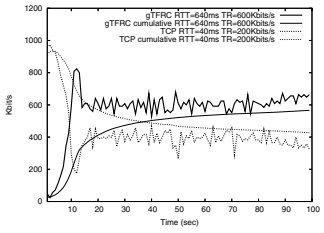
Figure 11. Over-provisioned network

4.4 Experiments versus TCP

This part looks at mixing one TCP flow and one TFRC or gTFRC flow in an over-provisioned network. Figures 12 show that the TCP flow is not disturbed by the gTFRC flow and remains the most aggressive in regard of its target rate. On both figures 12, TCP reaches a throughput about two times higher than its target rate. Figures 13 confirm this assumption. Both figures 13 seem identical, indeed, since the gTFRC flow has never been under its target rate during a long time, the TFRC standard algorithm is applied almost the whole experiment.

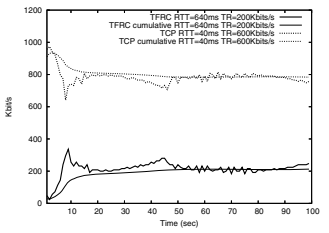


(a) TFRC versus TCP

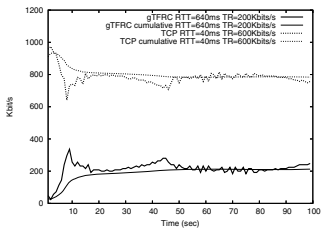


(b) gTFRC versus TCP

Figure 12. Over-provisioned network with TCP



(a) TFRC versus TCP



(b) gTFRC versus TCP

Figure 13. Over-provisioned network with TCP

4.5 Experiments with several flows

Finally, we conclude these experiments with one TCP flow versus 4 TFRC or gTFRC flow and one TFRC or gTFRC flow versus 4 TCP flows. Figures 14 and 15 give the results obtained. On scenario figure 14, TCP has a target rate of $400Kbits/s$ and each others four TFRC or gTFRC flows (respectively on figure 14 (a) and on figure 14 (a)) have a target rate of : $200Kbits/s, 100Kbits/s, 50Kbits/s, 50Kbits/s$. On figure 14, this is the TFRC or the gTFRC flow which has a target rate of $400Kbits/s$ and the four other TCP flows have a target rate of : $200Kbits/s, 100Kbits/s, 50Kbits/s, 50Kbits/s$. Concerning TCP alone, figure 14 (b) shows that the four gTFRC flows do not disturb the behavior of the TCP flow and that its behavior is similar with the TFRC experiment on figure 14 (a). Table 4 gives the throughput obtained by each flow and we can see that TCP obtains the same throughput with TFRC or gTFRC . In the reverse case, figures 15 show that the instantaneous throughput with gTFRC is better than TFRC and table 5 shows that whether TFRC or gTFRC flows obtain a similar average throughput. In these experiments, all flows are the same RTT of $40ms$. We have also made experiments with various RTT (confirmed by the previous experiments with two flows) and increased the number of flows and obtained similar results. For presentation sake, we give only the case of five flows.

5 Conclusions and further work

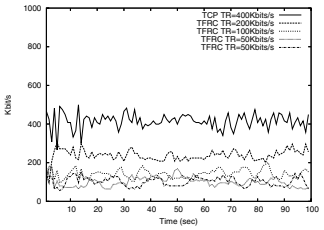
In this paper we proposed gTFRC : a simple and efficient adaption of TFRC congestion control for DiffServ network. gTFRC allows to reach a minimum guarantee throughput whether the RTT or the target rate of a flow. It requires only the target rate negotiated by the application in order to become QoS aware. We have demonstrated through many

	TCP TR=400Kb/s	TFRC or gTFRC TR=200Kb/s	TFRC or gTFRC TR=100Kb/s	TFRC or gTFRC TR=50Kb/s	TFRC or gTFRC TR=50Kb/s
TCP versus 4 TFRC	413.50	237.44	143.12	100	102.967
TCP versus 4 gTFRC	415.92	239.12	145.52	103.12	92.64

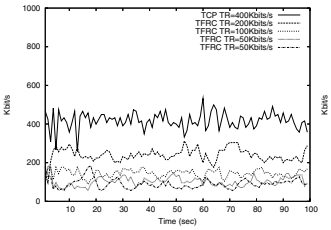
Table 4. Average throughput in Kbits/s for scenario with one TCP flow versus 4 TFRC or gTFRC

	TFRC or gTFRC TR=400Kb/s	TCP TR=200Kb/s	TCP TR=100Kb/s	TCP TR=50Kb/s	TCP TR=50Kb/s
TFRC versus 4 TCP	402.32	232.88	143.77	111.32	108.57
gTFRC versus 4 TCP	413.20	236.04	140.36	107.99	100.75

Table 5. Average throughput in Kbits/s for scenario with one TFRC or gTFRC flow versus 4 TCP

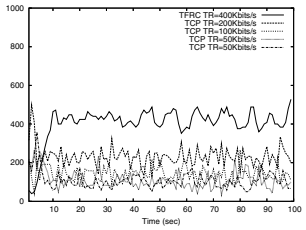


(a) One TCP versus 4 TFRC

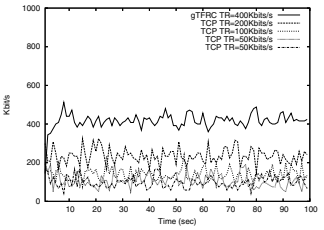


(b) One TCP versus 4 gTFRC

Figure 14. One TCP versus four TFRC and gTFRC (equal RTTs)



(a) One TFRC versus 4 TCP



(b) One gTFRC versus 4 TCP

Figure 15. One TFRC and gTFRC versus four TCPs (equal RTTs)

experiments its efficiency and that it can used in a standard AF/DiffServ class.

Future works are particularly devoted to the integration of gTFRC into the Enhanced Transport Protocol (ETP) [4]. ETP is a configurable protocol offering a partially ordered, partially reliable, congestion controlled and timed controlled end-to-end communication service suited for message-oriented multimedia flows in the context of QoS network. ETP, with gTFRC , will be evaluated over the pan-European EuQoS network. Secondly, we are currently studying the possibility to add gTFRC as a new CCID[1] into the DCCP protocol[15].

References

[1] C. Cicconetti, M. Garcia-Osma, X. Masip, J. Sa Silva, G. Santoro, G. Stea, and H. Taraskiuk. Simulation model for end-to-end QoS across heterogeneous networks. In *3rd International Workshop on Internet Performance, Simulation, Monitoring and Measurement (IPS-MoMe 2005)*, Warsaw, 2005.

[2] M. El-Gendy and K. Shin. Assured forwarding fairness using equation-based packet marking and packet separation. *Computer Networks*, 41(4):435–450, 2002.

[3] EuQoS. End-to-end quality of service support over heterogeneous networks. http://www.euqos.org/.

[4] E. Exposito. *Specifi cation and implementation of a QoS oriented Transport protocol for multimedia applications*. Phd thesis, LAAS-CNRS/ENSICA, Dec. 2003.

[5] W. Fang, N. Seddigh, and AL. A time sliding window three colour marker. Request For Comments 2859, IETF, June 2000.

[6] W. Feng, D. Kandlur, D. Saha, and K. S. Shin. Adaptive packet marking for providing differentiated services in the Internet. Technical Report CSE-TR-347-97, IBM, Oct. 1997.

[7] A. Feroz, A. Rao, and S. Kalyanaraman. A TCP-friendly traffi c marker for IP differentiated services. In *Proc. of IEEE/IFIP International Workshop on Quality of Service - IWQoS*, June 2000.

[8] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[9] M. Goyal, A. Durresi, R. Jain, and C. Liu. Effect of number of drop precedences in assured forwarding. In *Proc. of IEEE GLOBECOM*, pages 188–193, 1999.

[10] A. Habib, B. Bhargava, and S. Fahmy. A round trip time and time-out aware traffi c conditioner for differentiated services

[1] Congestion Control Identifi ers

networks. In *Proc. of the IEEE International Conference on Communications - ICC*, New-York, USA, Apr. 2002.

[11] M. Handley, S. Floyd, J. Pahdye, and J. Widmer. TCP Friendly rate control (TFRC): Protocol specifi cation. Technical Report 3448, IETF, Jan. 2003.

[12] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured forwarding PHB group. Request For Comments 2597, IETF, June 1999.

[13] J. Heinanen and R. Guerin. A single rate three color marker. Request For Comments 2697, IETF, Sept. 1999.

[14] Y.-G. Kim and C.-C. J. Kuo. TCP-Friendly assured forwarding (AF) video service in diffserv networks. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, Bangkok, Thailand, May 2003.

[15] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability, May 2003.

[16] K. Kumar, A. Ananda, and L. Jacob. A memory based approach for a TCP-friendly traffi c conditioner in diffserv networks. In *Proc. of the IEEE International Conference on Network Protocols - ICNP*, Riverside, California, USA, Nov. 2001.

[17] E. Lochin, P. Anelli, and S. Fdida. AIMD penalty shaper to enforce assured service for TCP fbws. In *4th International Conference on Networking (ICN'2005)*, La Reunion, France, Apr. 2005.

[18] E. Lochin, P. Anelli, and S. Fdida. Penalty shaper to enforce assured service for TCP fbws. In *IFIP Networking*, Waterloo, Canada, May 2005.

[19] N. Malouch and Z. Liu. Performance analysis of TCP with RIO routers. In *Proc. of IEEE GLOBECOM*, page 9, Taipei, Taiwan, Nov. 2002.

[20] B. Nandy, P.Pieda, and J. Ethridge. Intelligent traffi c conditioners for assured forwarding based differentiated services networks. In *IFIP High Performance Networking*, Paris, France, June 2000.

[21] E.-C. Park and C.-H. Choi. Proportional bandwidth allocation in diffserv networks. In *Proc. of IEEE INFOCOM*, Hong Kong, Mar. 2004.

[22] P. Pieda, J. Ethridge, M. Baines, and F. Shallwani. A network simulator differentiated services implementation.

[23] J. Postel. Transmission control protocol: Darpa internet program protocol specifi cation. Request For Comments 793, IETF, 1981.

[24] S. Sahu, P. Nain, C. Diot, V. Firoiu, and D. F. Towsley. On achievable service differentiation with token bucket marking for TCP. In *Measurement and Modeling of Computer Systems*, pages 23–33, 2000.

[25] N. Seddigh, B. Nandy, and P. Pieda. Bandwidth assurance issues for TCP fbws in a differentiated services network. In *Proc. of IEEE GLOBECOM*, page 6, Rio De Janeiro, Brazil, Dec. 1999.

[26] M. Singh, P. Pradhan, and P. Francis. Mpat: Aggregate tcp congestion management as a building block for internet qos. In *in Proceedings of IEEE International Conference on Network Protocols (ICNP 2004)*, Berlin, Germany, Oct. 2004.